

REPORT

THIẾT KẾ VÀ THỰC HIỆN HỆ THỐNG PHÁT HIỆN VÀ CẢNH BÁO RÒ RỈ KHÍ GAS TRONG NHÀ

	Họ và tên (Full name)	Mã SV (ID)	Đóng góp (Contribution)
Thành viên 1 (Member 1)	Vũ Minh Hoàng Tùng	22022107	35%
Thành viên 2 (Member 2)	Cao Song Toàn	22022188	65%
Tên/Địa chỉ Repo trên Github	https://github.com/toancst/Gas-Detection-Systems.git		

Tóm tắt (Abstract - from 5 to 10 lines)

Hệ thống phát hiện và cảnh báo rò rỉ khí gas được thiết kế để đảm bảo an toàn trong nhà ở hoặc môi trường dân dụng sử dụng khí đốt như LPG, methane. Dựa trên nền tảng vi điều khiển STM32F4, hệ thống thu thập tín hiệu từ cảm biến MQ2, tính toán nồng độ khí gas và hiển thị trên màn hình LCD. Khi khí vượt ngưỡng nguy hiểm, hệ thống kích hoạt còi báo động, LED cảnh báo và tự động ngắt van gas bằng rơ-le. Bên cạnh đó, một module ESP32 giao tiếp với STM32 để cung cấp khả năng giám sát từ xa qua web, cập nhật dữ liệu liên tục lên ThingSpeak và gửi thông báo đến người dùng qua WhatsApp. Người dùng có thể điều khiển hệ thống bằng nút nhấn vật lý hoặc từ xa thông qua giao diện web. Hệ thống hỗ trợ nhiều mức cảnh báo (an toàn, cảnh báo nhẹ, nguy hiểm) dựa trên giá trị ppm đo được. Thiết kế phần mềm nhúng đảm bảo phản ứng thời gian thực, dễ mở rộng và hiệu quả cho các ứng dụng thông minh. Đây là một giải pháp toàn diện giúp giảm thiểu nguy cơ cháy nổ do rò rỉ khí gas.

Từ khóa (Keywords)

IoT, Embedded System, MQ-2 Gas Sensors, Smart Home Safety,

Mục lục (Table of Contents)

Mục lục (Table of Contents)	3
1. Giới thiệu (Introduction)	4
2. Yêu cầu đối với thiết kế (Requirements)	5
2.1. Yêu cầu đối với thiết kế	5
2.2. Đặc tả kỹ thuật (Specification)	7
3. Thực hiện hệ thống (Implementation)	9
3.1. Kiến trúc phần cứng (Hardware Architecture)	9
3.1.1. Khối xử lý trung tâm	10
3.1.2. Cảm biến đầu vào	17
3.1.3. Các LED	17
3.1.4. Các nút bấm	18
3.1.5. LCD	18
3.2. Lập trình phần mềm	18
3.2.1. Hàm main() - Trung tâm vận hành	18
3.2.2. Nhóm hàm System (delay)	19
3.2.3. UART functions	19
3.2.4. GPIO - LED - BUTTON	19
3.2.5. ADC Function	19
3.2.6. Timer Function	19
3.2.7. Logic Control	19
3.2.8. Button Handling	20
3.2.9. PWM (Buzzer)	21
3.2.10. MQ2-Sensor	21
4. Kiểm chứng (Validation)	21
5. Kết luận (Conclusion)	22
Appendix A: Schematic	23
Appendix B: Code	24
References	25

1. Giới thiệu (Introduction)

Trong bối cảnh hiện nay, vấn đề an toàn trong các ngôi nhà thông minh ngày càng được quan tâm, đặc biệt là nguy cơ rò rỉ khí gas – một trong những nguyên nhân phổ biến gây cháy nổ nghiêm trọng. Việc phát hiện sớm và cảnh báo kịp thời có thể giúp ngăn chặn những hậu quả đáng tiếc về người và tài sản. Chính vì vậy, dự án này được thực hiện với mục tiêu xây dựng một hệ thống nhúng có khả năng giám sát, phát hiện và cảnh báo rò rỉ khí gas trong môi trường dân dụng.

Mục tiêu chính của dự án là vận dụng các kiến thức và kỹ năng đã học trong môn học hệ thống nhúng để thiết kế và triển khai một hệ thống có khả năng:

- Theo dõi liên tục nồng độ khí gas dễ cháy như LPG, methane, butane, propane trong không khí.
- Phát cảnh báo khi nồng độ khí vượt qua ngưỡng an toàn thông qua các phương tiện như đèn LED, còi buzzer hoặc hiển thị trên màn hình.
- Tự động ngắt nguồn cấp khí gas thông qua điều khiển van điện từ nhằm đảm bảo an toàn.
- Có khả năng mở rộng gửi thông báo đến người dùng qua kết nối không dây hoặc Internet.

Hệ thống được xây dựng trên nền tảng vi điều khiển STM32F4, sử dụng bo mạch STM32 Nucleo-F401RE hoặc tương đương (có lõi xử lý ARM Cortex-M). Các cảm biến và mô-đun ngoại vi như MQ2 để đo nồng độ khí gas, LCD để hiển thị thông tin, LED để biểu thị mức cảnh báo, buzzer để phát âm thanh, và van gas điều khiển bằng rơ-le đã được tích hợp vào hệ thống.

Thông qua việc hoàn thành dự án, sinh viên không chỉ củng cố kiến thức về cấu trúc phần cứng–phần mềm nhúng mà còn rèn luyện kỹ năng phân tích, thiết kế hệ thống thực tế và lập trình điều khiển các thiết bị ngoại vi. Đây là một bước chuẩn bị quan trọng cho các ứng dụng công nghiệp và dân dụng trong tương lai.

2. Yêu cầu đối với thiết kế (Requirements)

2.1. Yêu cầu đối với thiết kế

2.1.1. Các chức năng chính

Hệ thống phát hiện và cảnh báo rò rỉ khí gas trong nhà được xây dựng với mục tiêu đảm bảo an toàn bằng cách giám sát liên tục nồng độ khí gas trong môi trường và đưa ra các biện pháp cảnh báo, ngắt gas kịp thời. Các chức năng chính bao gồm:

a) Đo và xử lý dữ liệu từ cảm biến khí gas

Hệ thống sử dụng cảm biến MQ-2 để phát hiện nồng độ khí gas (LPG, CH₄, CO, v.v...).

Tín hiệu analog từ cảm biến được vi điều khiển STM32F4 đọc thông qua bộ chuyển đổi ADC tích hợp.

Sau khi đọc, giá trị ADC được xử lý để tính toán nồng độ khí gas tương đối (ppm) và được sử dụng để quyết định các hành động tiếp theo.

b) Phân loại mức nguy hiểm và hiển thị qua đèn LED

Dựa trên ngưỡng đã cấu hình, hệ thống phân loại mức độ rò rỉ khí thành 3 mức:

- An toàn: $\text{ppm} < 300 \rightarrow$ Bật LED xanh.
- Cảnh báo nhẹ: $300 \leq \text{ppm} < 900 \rightarrow$ Bật LED vàng.
- Nguy hiểm: $\text{ppm} \geq 900 \rightarrow$ Bật LED đỏ và nhấp nháy theo tần số 1 - 10 Hz phụ thuộc mức độ nghiêm trọng.

c) Kích hoạt cảnh báo âm thanh bằng buzzer

Khi nồng độ khí vượt quá ngưỡng > 1100 ppm, hệ thống tự động kích hoạt buzzer để phát ra âm thanh cảnh báo.

Tần số PWM điều khiển buzzer thay đổi để tạo hiệu ứng âm báo động rõ ràng.

d) Tự động đóng van gas khi phát hiện rò rỉ

Khi khí gas đạt ngưỡng nguy hiểm ($\text{ppm} > 1100$), hệ thống điều khiển relay đóng van gas, ngăn chặn nguy cơ cháy nổ.

e) Hiển thị thông tin trên màn hình LCD

Giá trị khí gas đo được và trạng thái cảnh báo hiện tại được hiển thị trên màn hình LCD giúp người dùng theo dõi dễ dàng.

f) Điều chỉnh hệ thống bằng nút nhấn vật lý

Nút RESET (PB0): Cho phép người dùng khởi động lại hệ thống nếu cần.

Nút STOP/CONTINUE (PB1): Tạm ngừng hệ thống trong các trường hợp đặc biệt và tiếp tục lại khi an toàn.

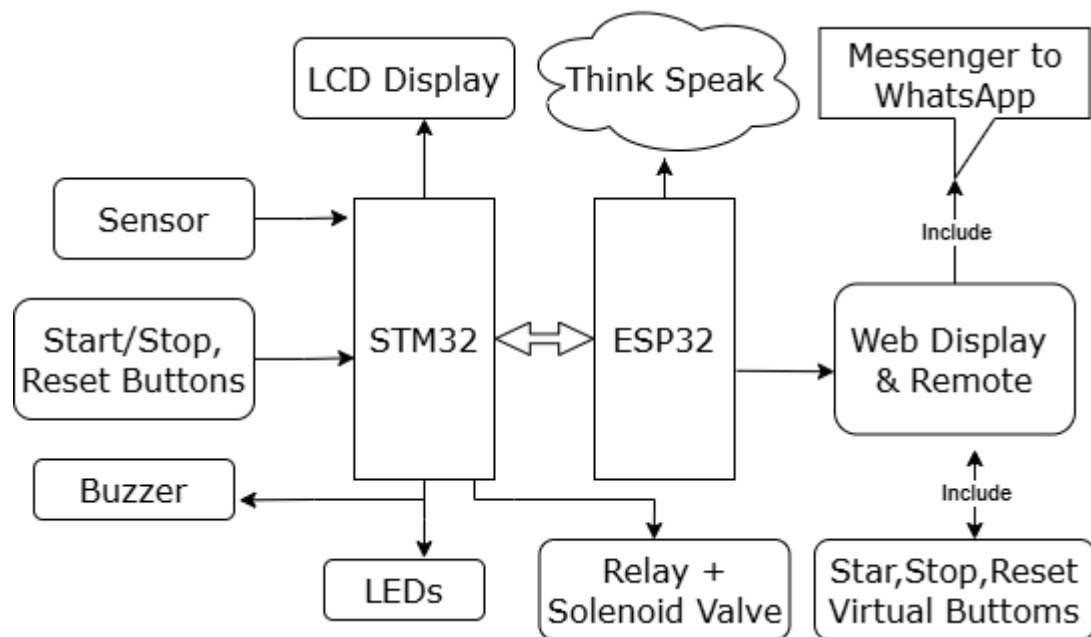
g) Giao tiếp và cảnh báo qua ESP32

STM32 truyền dữ liệu sang ESP32 qua UART.

ESP32 hoạt động như một web server hiển thị giao diện trạng thái hệ thống (giá trị ppm, trạng thái LED, van gas, v.v.), cũng có thể STOP/START, RESET hệ thống bằng các nút ảo trên web tương tự như khi bấm các nút vật lý.

Cập nhật và lưu các giá trị ppm theo thời gian liên tục lên ThinkSpeak. Khi khí gas vượt ngưỡng, ESP32 tự động gửi thông báo đến điện thoại người dùng qua WhatsApp Messenger, giúp phản ứng nhanh từ xa.

2.1.2 Sơ đồ khối chức năng:



Hình 1. Sơ đồ khối chức năng của hệ thống.

- *CPU*: vi điều khiển sử dụng vi xử lý ARM-Cortex-M (STM32 Nucleo-F401CCU6) thực hiện đọc cảm biến, xử lý ngưỡng và điều khiển các thiết bị cảnh báo.
- *Các đầu vào*:
 - o Hệ thống sử dụng cảm biến gas MQ-2 để phát hiện nồng độ khí gas ((LPG, CH₄, CO).

- SW1 để chuyển đổi hệ thống giữa trạng thái hoạt động và trạng thái dừng. Khi hệ thống dừng hoạt động, nếu bấm SW1 bộ đếm chuyển sang trạng thái hoạt động, và ngược lại. Khi chuyển từ trạng thái dừng sang trạng thái hoạt động, hệ thống tiếp tục giám sát trạng thái của cảm biến đo khí gas và hiển thị trên LCD.
- SW2 để xóa (reset) trạng thái hệ thống về trạng thái ban đầu.
- 2 nút bấm ảo tương tự SW1 và SW2 để điều chỉnh từ xa.
- *Các đầu ra:*
 - LED ba màu hiển thị 5 trạng thái:
 - LED xanh lá khi không có khí gas.
 - LED vàng khi nồng độ khí gas thấp.
 - LED đỏ nhấp nháy với tần số 1Hz khi nồng độ khí gas cao.
 - LED đỏ nháy ở tần số 2Hz-10Hz khi nồng độ khí cao trên mức nguy hiểm – tần số nháy càng cao khi nồng độ khí gas càng cao.
 - LED xanh dương khi hệ thống ở trạng thái dừng hoạt động.
 - Màn hình OLED/LCD: Hiển thị nồng độ gas, trạng thái hệ thống.
 - Hiển thị trạng thái của hệ thống: 0 – Không hoạt động, 1 – hoạt động
 - Trạng thái cảnh báo: 0 – không có khí gas, 1 – nồng độ khí gas thấp, 2- nồng độ khí gas cao, 3 – nồng độ khí gas cao trên mức nguy hiểm
- Buzzer: còi hú khi nồng độ khí gas vượt mức nguy hiểm khi ppm > 1100
- Rơ-le: điều khiển ngắt nguồn cung cấp gas khi phát hiện rò rỉ khi ppm > 1100.
- Sử dụng các timer để xác định thời gian nhấp nháy các LED.
- LCD display: Hiển thị thông tin thời gian thực từ STM32: giá trị ppm, trạng thái cảnh báo, trạng thái hệ thống.
- ESP32 Webserver:

Hiển thị trạng thái hệ thống (an toàn, báo động, nguy hiểm), cập nhật giá trị ppm liên tục mỗi 500ms.

Gửi thông báo về điện thoại qua WhatsApp mỗi khi giá trị ppm vượt các ngưỡng quy định.

2.2. Đặc tả kỹ thuật (Specification)

2.2.1. Tổng quan hệ thống

Hệ thống được thiết kế dựa trên kiến trúc hai vi điều khiển:

- Khối xử lý trung tâm (STM32): Chịu trách nhiệm thu thập dữ liệu từ cảm biến, xử lý thời gian thực và điều khiển trực tiếp các thiết bị cảnh báo cục bộ như đèn LED, còi và rơ-le ngắt van gas.
- Khối kết nối và giao tiếp (ESP32): Chịu trách nhiệm về các tác vụ kết nối không dây, cung cấp giao diện người dùng qua web, và gửi cảnh báo từ xa lên nền tảng IoT (ThingSpeak) và qua tin nhắn (WhatsApp).
- Hai khối này hoạt động song song, trong đó khối STM32 đảm bảo các phản ứng an toàn tức thời, còn khối ESP32 cung cấp khả năng giám sát và điều khiển từ xa.

2.2.2. Kiến trúc và chức năng các khối

a) Khối xử lý trung tâm

- Đầu vào :
 - Cảm biến khí gas (MQ-2)
 - Nút nhấn điều khiển
- Xử lý
 - Giá trị ADC được so sánh liên tục với các ngưỡng được định nghĩa trước để xác định mức độ nguy hiểm:
 - **GAS_THRESHOLD_LOW** (300): Ngưỡng an toàn.
 - **GAS_THRESHOLD_MID** (900): Ngưỡng cảnh báo.
 - **GAS_THRESHOLD_VALVE** (1100): Ngưỡng tự động đóng van gas.
 - **GAS_THRESHOLD_BUZZER** (1100): Ngưỡng kích hoạt còi báo động.
- Đầu ra :
 - Đèn LED trạng thái
 - Rơ-le van gas
 - Còi báo động
 - Giao tiếp UART với ESP32

b) Khối điều khiển giao tiếp

- Đầu vào : Dữ liệu từ STM32 giao tiếp qua UART
- Xử lý :
 - Kết nối vào mạng WiFi được cấu hình sẵn.
 - Khởi tạo một Web Server để cung cấp giao diện giám sát và điều khiển.

- Khởi tạo một Web Server để cung cấp giao diện giám sát và điều khiển.
- Đầu ra :
 - Giao diện Web giám sát
 - Cảnh báo từ xa (WhatsApp)

2.2.3. Luồng hoạt động chính

- a) **Khởi tạo** : Cả hai khối STM32 và ESP32 khởi tạo các ngoại vi, kết nối mạng (ESP32).
- b) **Giám sát** : Vòng lặp vô tận đọc giá trị ADC từ cảm biến MQ-2
- c) **Phân tích và cảnh báo cục bộ** : Dựa trên giá trị đọc được, STM32 điều khiển tức thời các đèn LED, còi và van gas theo các ngưỡng đã thiết lập.
- d) Giao tiếp và cảnh báo từ xa
- e) Tương tác người dùng

3. Thực hiện hệ thống (Implementation)

3.1. Kiến trúc phần cứng (Hardware Architecture)

Miêu tả chức năng của hệ thống (**Bảng 1 và Hình 1**);

Bảng 1: Mô tả các linh kiện được sử dụng.

STT	Tên ngoại vi	Chức năng
1	SW1	Start/Stop hệ thống
2	SW2	Reset hệ thống
3	MQ2-Sensor	Đo nồng độ khí gas trong môi trường. Giá trị được đọc thông qua ADC trong hàm ADC1_Read.
4	LCD Led display	Hiển thị trạng thái khí nồng độ khí gas
5	LED	Cảnh báo trực quan về mức độ nồng độ khí gas: Xanh (An toàn), Vàng (Cảnh báo), Đỏ nhấp nháy (Nguy hiểm). Logic điều khiển nằm trong hàm LED_Control

6	Buzzer	Phát cảnh báo bằng âm thanh khi nồng độ khí gas vượt ngưỡng nguy hiểm (GAS_THRESHOLD_BUZZER). Được điều khiển bằng PWM từ Timer 1
6	Gas Valve	Tự động đóng van gas để ngắt nguồn cung cấp khi nồng độ khí gas vượt ngưỡng an toàn (GAS_THRESHOLD_VALVE). Logic điều khiển trong hàm Gas_Valve_Control
7	ESP32 Module	Xử lý các tác vụ IoT: kết nối Wi-Fi, tạo giao diện web, gửi dữ liệu lên ThingSpeak để lưu trữ và gửi cảnh báo khẩn cấp qua WhatsApp

3.1.1. Khối xử lý trung tâm

a) Thuật toán chuyển đổi ADC, lấy mẫu và lọc nhiễu

ADC Analog to Digital Convert là bộ chuyển đổi tương tự sang số. Đại lượng tương tự là Điện áp Vin được so sánh với điện áp mẫu Vref (giá trị lớn nhất), sau đó được chuyển đổi thành số lưu vào thanh ghi DATA của bộ chuyển đổi đó.

Cảm biến MQ2 cho ra tín hiệu analog (điện áp liên tục từ 0-3.3V). Vậy nên ta cần phải sử dụng ADC để chuyển đổi điện áp analog thành số nguyên 12-bit.

MQ2 là cảm biến khí bán dẫn có khả năng phát hiện nhiều loại khí như LPG, CO, và khói. Cảm biến hoạt động dựa trên nguyên lý thay đổi điện trở khi tiếp xúc với các khí khác nhau.

Hệ thống áp dụng **bộ lọc trung bình động (Moving Average Filter)** để giảm nhiễu:

Công thức lọc trung bình:

$$Rs_avg = (1/N) \times \sum(Rs_i) \text{ với } i = 1 \text{ đến } N$$

=> Việc lấy trung bình của nhiều mẫu giúp giảm nhiễu ngẫu nhiên và tăng độ chính xác của phép đo.

Công thức chuyển đổi:

$$Rs = RL \times (Vcc - Vadc) / Vadc = RL \times (1023 - ADC_value) / ADC_value$$

Code:

```
float MQ2_MQResistanceCalculation(int raw_adc) {

    float flt_adc = (float) raw_adc;

    return RL_VALUE * (1023.0 - flt_adc) / flt_adc;

}
```

Trong đó:

- **Rs**: Điện trở cảm biến (Ω)
- **RL**: Điện trở tải (load resistor)
- **ADC_value**: Giá trị ADC đọc được (0-1023 cho 10-bit ADC)
- **1023**: Giá trị ADC tối đa ($2^{10} - 1$)

=> Cảm biến MQ2 và điện trở tải tạo thành mạch phân áp. Khi nồng độ khí thay đổi, điện trở cảm biến thay đổi, dẫn đến thay đổi điện áp tại điểm nối.

Ví dụ:

- **RL_VALUE = 5000 Ω**
- **ADC_value = 512** (tương ứng với $V_{cc}/2$)

$$R_s = 5000 \times (1023 - 512) / 512 = 5000 \times 511/512 \approx 4990\Omega$$

- Quy trình hiệu chuẩn (Calibration)

1. Tính toán R_o (điện trở trong không khí sạch)

$$R_o = R_{s_clean_air} / RO_CLEAN_AIR_FACTOR$$

Trong đó:

- **R_o** : Điện trở cảm biến trong không khí sạch
- **$R_{s_clean_air}$** : Điện trở đo được trong không khí sạch
- **$RO_CLEAN_AIR_FACTOR$** : Hệ số hiệu chuẩn từ datasheet (thường là 9.83)

Hiệu chuẩn là bước quan trọng để:

- Bù trừ sai số do điều kiện môi trường (nhiệt độ, độ ẩm)
- Chuẩn hóa cảm biến theo điều kiện tham chiếu
- Đảm bảo tính nhất quán giữa các cảm biến

- Chuyển đổi sang nồng độ khí (ppm):

Ta chuyển theo các bước sau:

$$R_s/R_o = k \times [\text{Gas}]^m$$

Trong đó k và m là hằng số đặc trưng cho từng loại khí

Biến đổi logarit:

$$\log(R_s/R_o) = \log(k) + m \times \log([\text{Gas}])$$

$$\log([\text{Gas}]) = (\log(R_s/R_o) - \log(k)) / m$$

$$\text{Nồng độ khí: } [\text{Gas}] = 10^{((\log(R_s/R_o) - \text{pcurve}[1]) / \text{pcurve}[2] + \text{pcurve}[0])}$$

- Thông số đường cong cho các khí khác nhau

// LPG Curve: [x, y, slope]

sensor->LPGCurve[0] = 2.3; // log10(200 ppm)

sensor->LPGCurve[1] = 0.21; // log10(Rs/Ro) tại 200 ppm

sensor->LPGCurve[2] = -0.47; // Độ dốc đường cong

// CO Curve

sensor->COCurve[0] = 2.3;

sensor->COCurve[1] = 0.72;

sensor->COCurve[2] = -0.34;

// Smoke Curve

sensor->SmokeCurve[0] = 2.3;

sensor->SmokeCurve[1] = 0.53;

sensor->SmokeCurve[2] = -0.44;

Ví dụ:

- $R_s = 3000\Omega$
- $R_o = 5000\Omega$

- Sử dụng LPG curve

$$R_s/R_o = 3000/5000 = 0.6$$

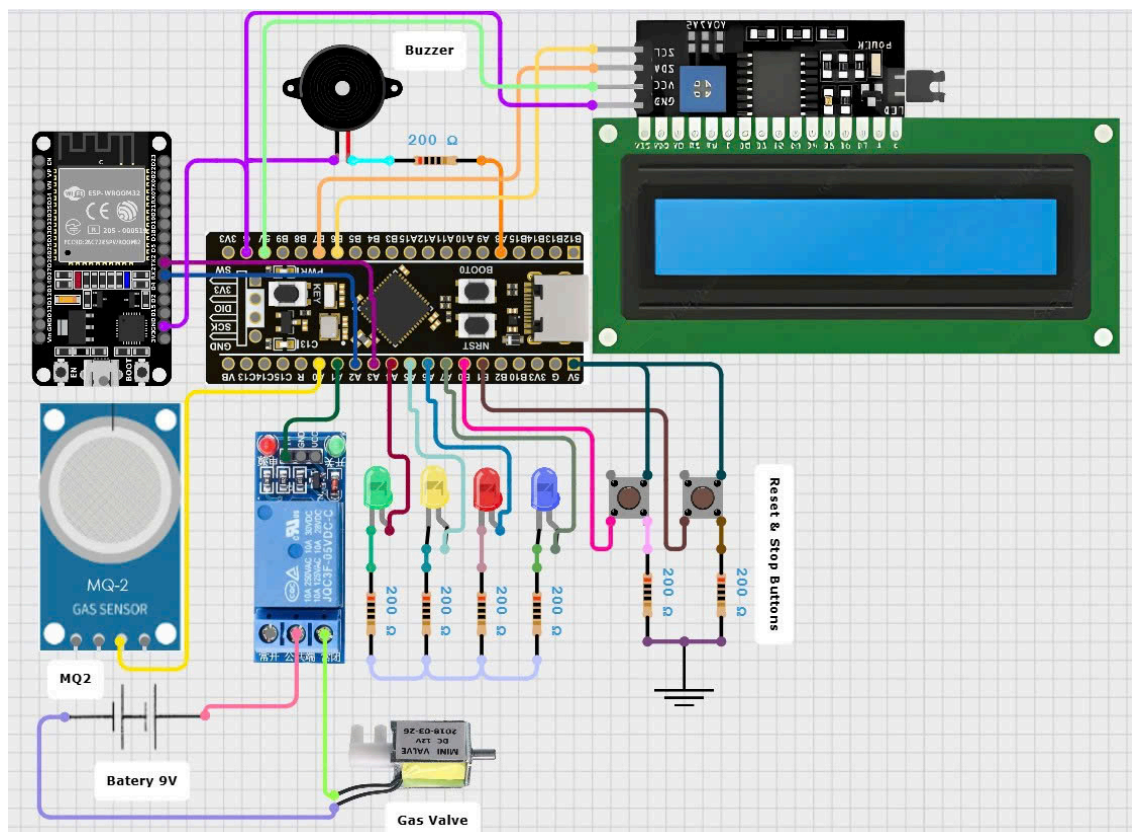
$$\log(R_s/R_o) = \log(0.6) = -0.222$$

$$[LPG] = 10^{((-0.222 - 0.21) / (-0.47) + 2.3)}$$

$$= 10^{((-0.432) / (-0.47) + 2.3)}$$

$$= 10^{(0.919 + 2.3)}$$

$$\approx 1656 \text{ ppm}$$



b) LED nhấp theo tần số

Tần số nhấp nhấp tăng theo nồng độ khí gas: Nhấp nhấp chậm = ít nguy hiểm, nhấp nhấp nhanh = rất nguy hiểm.

Thành phần chính của timer chính là bộ đếm – counter (CNT), với các ngưỡng trên được thiết lập bởi thanh ghi Auto Reload (ARR). Counter có thể đếm lên hoặc đếm xuống. Clock đưa vào bộ đếm có thể được chia bởi một bộ chia tần – Prescaler.

Người dùng có thể thực hiện các lệnh đọc, ghi vào các thanh ghi CNT, ARR và PSC để cấu hình cho khối cơ sở của mỗi bộ Timer.

– Counter Register (TIMx_CNT): Khi hoạt động, thanh ghi này tăng hoặc giảm giá trị theo mỗi xung clock đầu vào. Tùy vào bộ timer mà counter này có thể là 16bit hoặc 32bit.

– Prescaler Register (TIMx_PSC): Giá trị của thanh ghi bộ chia tần (16bit) cho phép người dùng cấu hình chia tần số đầu vào (CK_PSC) cho bất kì giá trị nào từ [1-65536]. Sử dụng kết hợp bộ chia tần của timer và của RCC giúp chúng ta có thể thay đổi được thời gian của mỗi lần CNT thực hiện đếm, giúp tạo ra được những khoảng thời gian, điều chế được độ rộng xung phù hợp với nhu cầu.

– Auto-Reload Register (TIMx_ARR): Giá trị của ARR được người dùng xác định sẵn khi cài đặt bộ timer, làm cơ sở cho CNT thực hiện nạp lại giá trị đếm mỗi khi tràn (overflow khi đếm lên – CNT vượt giá trị ARR, underflow khi đếm xuống – CNT bé hơn 0). Tùy vào bộ timer mà counter này có thể là 16bit hoặc 32bit.

Ở đây với tần số hệ thống (System Clock) là 84 MHz.

- Prescaler = 8400 → mỗi lần đếm mất 100 μ s.
- ARR = 10 → sau 10 lần đếm → 1 ms.

=> Ngắt Timer xảy ra mỗi 1 ms.

- Công thức chia các khoảng tần số:

$$\text{range} = \text{THRESHOLD_HIGH} - \text{THRESHOLD_MID} = 1600 - 900 = 700$$

$$\text{step} = \text{range} / 10 = 700 / 10 = 70$$

$$f_LED = ((\text{Gas_Value} - 900) / 70) + 1$$

- Ví dụ:

Gas_Value = 900 \rightarrow $f_{LED} = ((900-900)/70) + 1 = 1$ Hz (chậm)

Gas_Value = 1250 \rightarrow $f_{LED} = ((1250-900)/70) + 1 = 6$ Hz (trung bình)

Gas_Value = 1600 \rightarrow $f_{LED} = ((1600-900)/70) + 1 = 11$ Hz \rightarrow clamp về 10 Hz (nh nhanh nhất)

- Tính toán chu kỳ timer cho LED:

Period_ticks = Base_Period / Frequency = 50 / f_{LED}

- Ta có:

- Timer ngắt mỗi 10ms (100 Hz)
- Để có 1 Hz cần: 1000ms / 10ms = 100 ticks
- Nhưng LED toggle (đảo trạng thái) nên cần: 100/2 = 50 ticks
- 1 Hz = 50 ticks, 2 Hz = 25 ticks, 10 Hz = 5 ticks

Suy ra:

- Prescaler = 8400 \rightarrow mỗi lần đếm mất **100 μ s**.
- ARR = 10 \rightarrow sau 10 lần đếm \rightarrow 1 ms.

- Ví dụ:

$f_{LED} = 2$ Hz \rightarrow Period = 50/2 = 25 ticks = 250ms ON + 250ms OFF = 0.5s/cycle

$f_{LED} = 5$ Hz \rightarrow Period = 50/5 = 10 ticks = 100ms ON + 100ms OFF = 0.2s/cycle

$f_{LED} = 10$ Hz \rightarrow Period = 50/10 = 5 ticks = 50ms ON + 50ms OFF = 0.1s/cycle

c) PWM Buzzer

Buzzer chỉ nhận 1 giá trị 5v liên tục sẽ nghe đơn điệu, không thu hút sự chú ý. \Rightarrow Cần phải tạo ra tiếng báo động như còi hú như của các xe chữa cháy.

Thực hiện bằng cách tạo âm thanh biến thiên bằng cách kết hợp 2 dạng sóng:

Sóng tam giác - tạo độ biến thiên:

$$T(t) = \left\{ \begin{array}{ll} 2t, & \text{nếu } 0 \leq t < 0.5 \quad (\text{tăng dần}) \\ 2(1-t), & \text{nếu } 0.5 \leq t \leq 1 \quad (\text{giảm dần}) \end{array} \right\}$$

Sóng sine - tạo độ mượt mà:

$$S(t) = (\sin(2\pi t) + 1) / 2$$

(+1 để dịch từ [-1,1] lên [0,2]

/2 để chuẩn hóa về [0,1])

Sóng kết hợp:

$$\text{PWM}(t) = T(t) \times S(t) \times 999$$

=> **Kết quả:** Âm thanh có độ cao thay đổi từ 0→499→499→0, lặp lại tạo hiệu ứng "warbling".

- Cấu hình PWM:

$$f_{\text{PWM}} = f_{\text{TIM}} / ((\text{PSC}+1) \times (\text{ARR}+1))$$

$$1000 = 84000000 / (84 \times 1000)$$

- Chọn PWM Mode 1:

$\text{TIMx_CNT} < \text{TIMx_CCR1}$: Output HIGH

$\text{TIMx_CNT} \geq \text{TIMx_CCR1}$: Output LOW

$$\text{Duty cycle} = \text{CCR1}/\text{ARR} \times 100\%$$

Ví dụ:

- $\text{CCR1} = 0 \rightarrow \text{Duty} = 0\% \rightarrow \text{Buzzer OFF}$
 - $\text{CCR1} = 500 \rightarrow \text{Duty} = 50\% \rightarrow \text{Âm thanh vừa}$
 - $\text{CCR1} = 999 \rightarrow \text{Duty} = 99.9\% \rightarrow \text{Âm thanh to nhất}$
- d) Xử lý nút nhấn và chống dội

Khi nhấn nút cơ học, tiếp điểm đóng-mở nhiều lần trong vài ms đầu dẫn đến vi điều khiển đọc như nhiều lần nhấn thay vì 1 lần.

Sử dụng 2 phương pháp Edge Detection + Software Debouncing để xử lý vấn đề trên

Công thức phát hiện cạnh lên tại chân đọc nút bấm:

$$\text{Rising_Edge} = \text{Current_State} \wedge (\neg \text{Previous_State})$$

$$\text{Rising_Edge} = \text{Bt}(n) \wedge (\neg \text{Bt}(n-1))$$

- Ví dụ:

$$\text{Previous} = 0, \text{Current} = 0 \rightarrow \text{Edge} = 0 \wedge 1 = 0 \text{ (không có cạnh)}$$

$$\text{Previous} = 0, \text{Current} = 1 \rightarrow \text{Edge} = 1 \wedge 1 = 1 \text{ (có cạnh lên!)}$$

$$\text{Previous} = 1, \text{Current} = 1 \rightarrow \text{Edge} = 1 \wedge 0 = 0 \text{ (không có cạnh)}$$

$$\text{Previous} = 1, \text{Current} = 0 \rightarrow \text{Edge} = 0 \wedge 0 = 0 \text{ (cạnh xuống, không quan tâm)}$$

- Cấu hình Pull-down resistor:

GPIOB->PUPDR |= (2 << 0); // PB0 pull-down

Không có pull-down: Chân floating, đọc giá trị ngẫu nhiên

Có pull-down: Chân = 0V khi không nhấn, = 3.3V khi nhấn

Pull-down vs Pull-up: Pull-down cho logic positive (nhấn = 1), dễ hiểu hơn

e) Thuật toán điều khiển van gas

Nếu gas value dao động quanh ngưỡng (999↔1001), van sẽ đóng-mở liên tục.

=> Sử dụng hysteresis với 2 ngưỡng khác nhau:

Open_Threshold = 1000 - Hysteresis = 950

Close_Threshold = 1000

- Logic điều khiển:

```
if (Gas_Value > 1000 && Valve_State == OPEN) {  
    Valve_State = CLOSE;  
}  
  
if (Gas_Value < 950 && Valve_State == CLOSE) {  
    Valve_State = OPEN;  
}
```

- Ví dụ:

Gas = 800 → Van MỞ (an toàn)

Gas = 1100 → Van ĐÓNG (nguy hiểm, lần đầu vượt 1000)

Gas = 980 → Van vẫn ĐÓNG (chưa xuống dưới 950)

Gas = 900 → Van MỞ (đã xuống dưới 950, an toàn trở lại)

3.1.2. Cảm biến đầu vào

MQ2 AO → STM32 PA0 (ADC input)

VCC → 3.3V

GND → GND

3.1.3. Các LED

Hệ thống sử dụng 4 LED gồm đỏ, vàng, xanh lá và LED dừng (stop) để hiển thị mức độ nguy hiểm của khí gas và trạng thái hệ thống. Các LED được nối với các chân trên cổng GPIOA như sau:

Bảng 2. Kết nối các LED trên bo mạch STM32

LED	Pin STM32
Red	PA6
Yellow	PA5
Green	PA4
Stop (Blue)	PA7

LED đỏ sẽ nhấp nháy với tần số tỉ lệ thuận với mức độ khí gas, LED vàng báo mức trung bình, LED xanh báo an toàn, và LED xanh dương bật khi hệ thống chuyển sang chế độ "Stop".

3.1.4. Các nút bấm

Hệ thống sử dụng hai nút bấm:

- **PB0 (GPIOB pin 0):** Nút reset hệ thống. Khi được nhấn, hệ thống gửi chuỗi "Reset" qua UART và thực hiện reset bằng `NVIC_SystemReset()`.
- **PB1 (GPIOB pin 1):** Nút chuyển chế độ Stop/Continue. Nhấn một lần để đưa hệ thống vào chế độ dừng, nhấn lại để tiếp tục vận hành.

Các nút được cấu hình ở chế độ input pull-down, kiểm tra trạng thái trong vòng lặp `while(1)` và có bảo vệ bằng timer để tránh nhấn nhầm hoặc nhấn quá nhanh.

3.1.5. LCD

LCD giao tiếp qua giao thức I2C sử dụng chân GPIOB6 (SCL) và GPIOB7 (SDA). LCD được sử dụng để hiển thị trạng thái khởi tạo, hiệu chuẩn cảm biến, và giá trị nồng độ khí gas hiện tại.

Một số hàm điều khiển LCD bao gồm: `LCD_Init()`, `LCD_Print()`, `LCD_SetCursor()`, và `LCD_Update_Display()`.

3.2. Lập trình phần mềm

Dưới đây là bài giải thích chi tiết cho từng nhóm hàm trong file `main.c` trong hệ thống cảnh báo gas sử dụng STM32F4 và cảm biến MQ2:

3.2.1. Hàm `main()` - Trung tâm vận hành chương trình

Hàm `main()` chịu trách nhiệm khởi tạo toàn bộ hệ thống và vòng lặp chính:

- Khởi tạo clock và các ngoại vi: UART, ADC, LED, nút nhấn, van gas, Timer, LCD.
- Khởi tạo PWM và bảng tính PWM cho buzzer.
- Khởi tạo và hiệu chuẩn MQ2, hiển thị trạng thái lên LCD.
- Vòng lặp vô hạn:
 - Đọc ADC (giá trị gas), gửi UART, hiển thị LCD.
 - Xử lý LED, buzzer, van gas tùy theo ngưỡng gas.
 - Kiểm tra và xử lý nhấn nút PB0 (Reset) và PB1 (Stop/Resume).

3.2.2. Nhóm hàm `System (delay)`

- `delay_ms(uint32_t ms)`
 - Tạo delay theo millisecond dùng SysTick.
- Chu kỳ 1ms = 16000 chu kỳ với HCLK = 16 MHz.
- `delay_us(uint16_t us)`
 - Dùng counter của TIM1 để delay theo micro giây.

3.2.3. UART Functions

`UART2_Init()` : Khởi tạo USART2 (TX = PA2, RX = PA3) với baudrate ~115200.

`UART2_SendChar(uint8_t c)` : Gửi một byte qua USART2.

`UART2_SendString(char *str)` : Gửi chuỗi char cho đến khi gặp '\0'.

3.2.4. GPIO - LED - BUTTON

LED_Init() : Cài đặt LED Xanh (PA4), Vàng (PA5), Đỏ (PA6), STOP (PA7) là output.

Gas_Valve_Init() : PA1 làm output điều khiển van gas (1: đóng, 0: mở).

Button_Init() : PB0, PB1 là input pull-down.

Read_Button(GPIO_TypeDef*, uint16_t) : Trả về 1 hoặc 0 tùy theo trạng thái chân.

3.2.5. ADC Function

ADC1_Init() : Bật ADC1, cấu hình GPIOA0 (chân MQ2) là analog.

Tạo delay ngắn sau khi bật ADC.

ADC1_Read() : Khởi động chuyển đổi ADC, chờ hoàn thành và đọc kết quả 12-bit.

3.2.6. Timer Function

TIM1_Init() : Sử dụng cho delay us (Prescaler = 15, ARR = 0xFFFF).

TIM2_Init() : Dùng tạo ngắt mỗi 1ms (Prescaler 8400-1, ARR = 10). Bật ngắt và đặt độ ưu tiên cao nhất.

TIM2_IRQHandler() : Tăng counter toàn cục.

3.2.7. Logic Control

LED_Control(gasValue) Hiển thị LED:

- Gas < 300: Bật xanh.
- 300 < GAS < 900: Vàng.
- GAS > 900: nhấp nháy LED đỏ theo tần số.

Calculate_Red_LED_Frequency(gasValue) : Tính tần số nhấp nháy LED đỏ theo gas.

Gas_Valve_Control(gasValue) : Đóng van gas nếu gas > 1000, mở nếu thấp hơn.

Send_Gas_Value(gasValue) : Gửi giá trị ADC qua UART (dạng chuỗi).

Get_Gas_Alert_Level(gasValue) : Trả về mức độ nguy hiểm (0-3).

3.2.8 Button Handling

Handle_PB0_Press() Nhấn PB0 (Reset) khi không bị protection: gọi System_Reset().

Handle_PB1_Press()

- Nhấn PB1:
 - Chưa vào stop mode: gọi Enter_Stop_Mode().

- Trong stop mode và không bị protection: Exit_Stop_Mode().

System_Reset() : Gửi UART "Reset" rồi gọi NVIC_SystemReset().

Enter_Stop_Mode() : Vô hiệu hóa hoạt động, tắt LED + buzzer, đồng van gas.

Exit_Stop_Mode() : Khôi phục vận hành bình thường.

3.2.9. PWM (Buzzer)

make_triangle_sin_table() : Tạo bảng giá trị PWM dạng tam giác nhân sin.

init_pwm() : Khởi tạo TIM1 CH1 (PA8) làm PWM.

update_pwm() : Nếu buzzer_active, cập nhật giá trị PWM từ bảng.

3.2.10. MQ-2 Sensor

MQ2_create(MQ2 *sensor) : Gán đường cong (curve) cho LPG, CO, Smoke.

MQ2_begin()

Hiệu chuẩn: tính Rs trong clean air, chia cho RO_CLEAN_AIR_FACTOR → ra Ro.

MQ2_checkCalibration() : Kiểm tra Ro hợp lệ hay chưa.

MQ2_MQResistanceCalculation(int adc) : Tính $R_s = R_L * (4095 - ADC) / ADC$.

MQ2_MQCalibration() : Lặp nhiều lần để tính Rs trung bình.

MQ2_MQRead() : Tính Rs trung bình khi hoạt động bình thường.

MQ2_MQGetPercentage(float *pcurve, MQ2 *sensor) : Tính ppm theo công thức logarit.

MQ2_read(MQ2 *sensor, bool print) : Đọc cả 3 giá trị (LPG, CO, Smoke), gửi UART nếu print = true.


MQ2_readLPG() / MQ2_readCO() / MQ2_readSmoke() : Trả về giá trị ppm tương ứng với bộ nhớ cache hoặc đọc mới.

4. Kiểm chứng (Validation)

Hệ thống được kiểm thử theo các nội dung chính sau:

- **Khởi động hệ thống:** LCD hiển thị thông báo khởi tạo và hiệu chuẩn cảm biến; UART in ra chuỗi xác nhận "**System starting...**" và giá trị Ro sau hiệu chuẩn.
- **Đọc và xử lý giá trị khí gas:** ADC hoạt động ổn định, giá trị đo được hiển thị chính xác trên LCD và gửi qua UART định kỳ.
- **Cảnh báo bằng LED và buzzer:** LED xanh, vàng và đỏ thay đổi đúng theo mức nồng độ khí gas. Buzzer phát âm cảnh báo khi nồng độ vượt ngưỡng nguy hiểm.
- **Điều khiển van gas:** Van gas đóng lại khi phát hiện khí vượt 1000 ppm và mở lại khi nồng độ giảm.
- **Tương tác người dùng:** Nút PB0 cho phép reset hệ thống, nút PB1 chuyển đổi giữa chế độ hoạt động và chế độ dừng. Cả hai nút đều phản hồi đúng chức năng.
- **Hiển thị LCD:** Màn hình cập nhật giá trị khí mỗi 500ms và phản ánh chính xác trạng thái hệ thống.

Tất cả các chức năng đều được kiểm thử và hoạt động ổn định trong các điều kiện thực nghiệm. Kết quả kiểm chứng được ghi lại bằng video thực hiện bởi nhóm sinh viên Cao Song Toàn và Vũ Minh Hoàng Tùng.

Video :  Nhóm: Cao Song Toàn - Vũ Minh Hoàng Tùng

5. Kết luận (Conclusion)

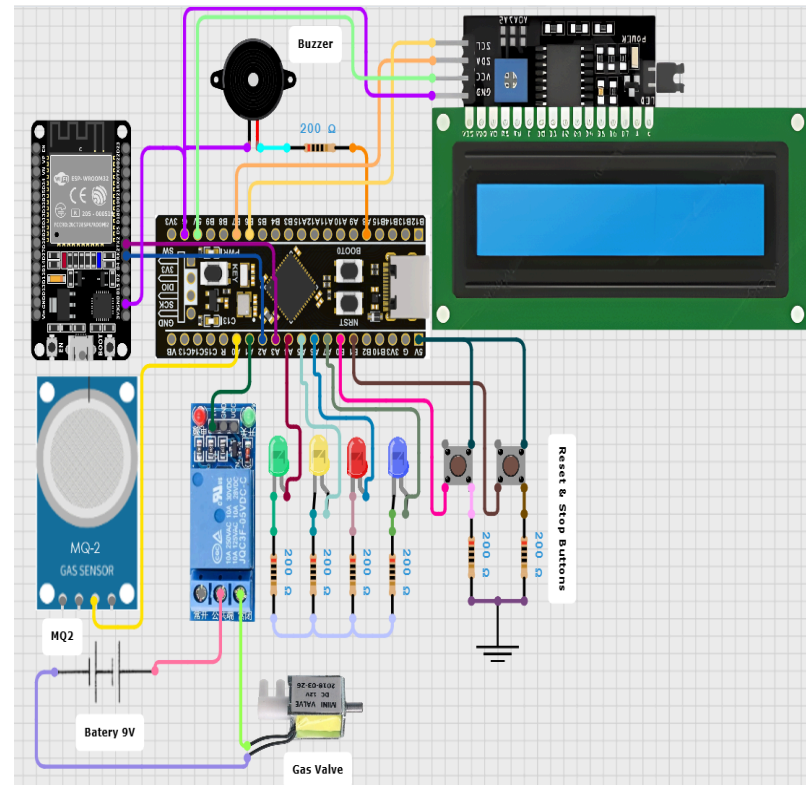
Qua quá trình thực hiện đề tài, nhóm đã xây dựng thành công một hệ thống cảnh báo rò rỉ khí gas sử dụng vi điều khiển STM32F4 và cảm biến MQ2. Hệ thống có khả năng đo lường nồng độ khí gas trong không khí, xử lý tín hiệu theo ngưỡng cảnh báo đã định sẵn và phản hồi bằng các phương tiện cảnh báo như đèn LED, còi báo động, hiển thị LCD và điều khiển van gas. Tất cả các chức năng đều được tích hợp trong một chương trình điều khiển trung tâm và hoạt động ổn định sau khi kiểm thử thực tế.

Việc hoàn thiện hệ thống giúp nhóm hiểu rõ hơn về cách lập trình các ngoại vi như ADC, GPIO, UART, PWM, Timer trên vi điều khiển STM32, đồng thời rèn luyện kỹ năng thiết kế phần mềm nhúng theo cấu trúc modul hóa và dễ mở rộng. Bên cạnh đó, nhóm cũng học được cách xử lý tín hiệu cảm biến, hiệu chuẩn phần cứng, quản lý trạng thái hệ thống theo thời gian thực và tương tác với người dùng qua giao diện đơn giản.

Mặc dù hệ thống đã đáp ứng đầy đủ các yêu cầu cơ bản của đề tài, vẫn còn nhiều tiềm năng phát triển trong tương lai. Một số hướng mở rộng bao gồm: tích hợp kết nối không dây (Wi-Fi, Bluetooth) để giám sát từ xa, đưa dữ liệu lên nền tảng đám mây hoặc ứng dụng di động; bổ sung bộ nhớ lưu trữ để theo dõi lịch sử khí gas; và cải thiện cơ chế cảnh báo như gửi tin nhắn SMS hoặc tích hợp với hệ thống báo cháy. Việc cải tiến giao diện người dùng hoặc thêm các cảm biến bổ sung (ví dụ: nhiệt độ, độ ẩm) cũng là những hướng đi khả thi giúp nâng cao độ chính xác và độ tin cậy của hệ thống.

Tổng kết lại, quá trình thực hiện bài tập lớn không chỉ giúp nhóm hoàn thiện một sản phẩm kỹ thuật có tính ứng dụng thực tiễn cao mà còn mang lại nhiều kinh nghiệm quý báu trong thiết kế hệ thống nhúng, làm việc nhóm, và giải quyết vấn đề kỹ thuật trong môi trường hạn chế tài nguyên.

Appendix A: Schematic



Appendix B: Code

Mã nguồn của dự án : <https://github.com/toancst/Gas-Detection-Systems>

References

- [1] [ESP32 Web Server with BME680 - Weather Station | Random Nerd Tutorials](#)
- [2] [ESP32 - Gas Sensor | ESP32 Tutorial](#)
- [3] [Ifarul/STM32-gas-sensor-and-temperature-measurement](#)
- [4] [MQ-2 Datasheet\(PDF\) - Zhengzhou Winsen Electronics Technology Co., Ltd.](#)
- [5] [labay11/MQ-2-sensor-library: A simple library to retrieve the information given from the MQ 2 sensor in arduino](#)
- [6] [MQ_2 Sensor Callibration - File Exchange - MATLAB Central](#)
- [7] [Embedded Systems Tutorial](#)
- [8] [Getting Started with STM32 - Timers and Timer Interrupts](#)